디지털 사회
DIGITAL SOCIETY
제9호

연세대학교 사회과학데이터혁신연구센터
SOCIAL SCIENCE DATA INNOVATION CENTER

발 행 인: 조화순
발 행 일: 2018년 1월 15일
홈페이지: http://democracy3.0.yonsei.ac.kr
ISSN 2586-3525(Online)

코딩과 어학의 공통점

강정한
연세대학교 사회학과 교수

요즘 빅데이터의 중요성, 4차 산업혁명의 도래, 취업난 심화 등의 여파로 코딩 교육 열풍이 불고 있다. 일부 대학들은 코딩을 필수 교양으로 선정하고 있고, 조만간 초등교육부터 전면 도입될 수도 있으며, 코딩 사교육까지 활성화되고 있다. 그러나 그간의 삶이 코딩과 무관했던 사람이라면 이러한 코딩 교육에 능동적으로 동참하기 어렵고, 의욕적으로 참여하더라도 맹목적이거나 잘못된 방향으로 코딩을 배우기 쉽다. 필자 역시 코딩을 배우고 있는 입장에서 겪는 시행착오와 잘 배우는 사람에 대한 관찰을 바탕으로 코딩을 어떤 관점과 태도로 배우거나 이해하는 것이 바람직할지 이 글을 통해 공유해보려 한다.

프로그래머가 될 것도 아닌데 코딩을 꼭 배워야 할까? 그런데 우리는 영문학자가 될 것도 아니지만 영어는 모두 배운다. 급변하는 사회에서 코딩이 필요한 이유는 영어가 필요한 이유와 비슷하다. 코딩은 영어처럼 하고 싶은 일을 하기 위한 소통 수단이 되어 가고 있다. 따라서 효과적으로 코딩을 배우는 방법은 효과적인 어학 공부 방법과 비슷해야 한다. 그러나 현재 불고 있는 코딩 열풍은 프로그래머 교육과 큰 차별이 없어, 영문법 책만 파면서 생활영어를 배우고자 하는 것과 비슷하다. 바꾸어 말하면 코딩교육이 '프로그래밍' '언어'를 습득하는 것이라고 했을 때, '프로그래밍'의 습득보다는 '언어' 습득에 중점을 두고 접근하는 것이 필요하다.

코딩의 필요성에 관해 필자가 가장 인상 깊게 들은 강연은 미치 레스닉(Mitch Resnick)이라는 MIT 교수가 테드(TED) 강연에서 한 말이다. 그 교수에 따르면 우리가 "read to learn"를 위해 "learn to read"를 먼저 해야 하듯이, "code to learn"을 위해 "learn to code"를 해야 한다는 것이다. 즉 코딩을 통해 세상을 배울 필요성이 증가하기 때문에 코딩부터 배워야 한다는 것을 분명히 하고 있다. 이러한 관점은 코딩이 전문가로서 프로그래밍 기술이라기보다 우리의 삶에 필요한 교양 언어의 역할을 함을 분명히 하고 있다. 우리는 커리어를 쌓기 위한 이력서에 종종 자신의 어학 실력을 언어별로 고급, 중급 등으로 표시하곤 한다. 프로그래밍 언어인 Python, Java 등을 영어, 일어 등 외국어와 함께 어학 실력으로 묶어 이력서에 표기하는 것이 자연스러운 날이 올 수 있다.

우리는 흔히 실전 영어와 교실 영어를 구분하곤 한다. 즉, 아무리 영어 문법책을 많이 공부하고 단어를 많이 알아도 실생활에서 영어를 사용할 기회나 필요성이 없으면 영어를 제대로 활용하지 못한다. 이는 코딩에서도 마찬가지다. 특정 프로그래밍 언어에 대해 좋은 교과서로 성실히 배워도 그 언어를 실제로 사용할 일이 없다면 코딩 능력은 떨어진다. 영어를 이용해 카페에서 주문을 하고 친구와 대화를 할 때 영어 실력이 늘듯이, 코딩을 이용해 내가 원하는 무엇인가를 해야 할 때 코딩 실력이 는다. 즉 문제해결형 활동에 코딩이 실제로 동원되지 않는다면 아무리 좋은 교과서, 좋은 강좌로 코딩을 배워도 코딩이 손에 익지 않는다.

어학 실력은 흔히 계단식으로 발전한다. 말하기건 쓰기건 한동안은 들인 노력에 비해 가시적 발전이 없다가 한 순간 도약하고 다시 정체되곤 한다. 코딩도 비슷하다. 초기에는 감도 잡히지 않고 헤메는 기간이 길다가 어느 순간 늘어난 실력이 보인다. 특히 자신이 짠 코드가 의도대로 작동할 때의 성취감은 카페에서 영어로 주문한 음료가 올바로 나왔을 때의 안도감보다 크기 때문에, 이러한 발전의 단계는 더 분명히 느껴지고 보람이 있다. 한편 첫 번째 발전의 단계가 빨리 나타나지 않기 때문에 심리적 진입장벽이 높고 코딩을 시작하기 어려울 수도 있다. 그러나 필자도 코딩을 배우는 중이라 확신할 수는 없지만, 우리가 영어를 배우는데 얼마나 많은 세월을 투자했는지 돌이켜 본다면, 코딩으로 특정 발전 단계에 도달하는 시간은 영어와 비교도 안 될 정도로 짧다.

내성적인 사람이 외국어를 빨리 배울까, 사교적인 사람이 외국어를 빨리 배울까? 언어는 소통수단이기 때문에 당연히 후자다. 그렇다면 내성적인 사람과 사교적인 사람 중 누가 코딩을 빨리 배울까? 보통 컴퓨터 전문가들은 사회생활 없이 혼자 모니터와만 대화하는 과장된 이미지로 비춰지기도 하지만, 이러한 이미지는 적어도 교양 언어로서 코딩이 필요하고 배우는 경우에는 해당하지 않는다. 혼자서 코딩 에러를 해결하려고 끙끙대기보다는 주변과 소통하면서 도움을 주고받으며 문제를 해결하는 사람이 빨리 배운다. 여기서 말하는 소통을 통한 문제해결은 꼭 면대면 소통만을 뜻하지 않는다. stackoverflow같은 온라인 공동체를 잘 이용할 줄 아는 소통도 중요하다. 그리고 코딩을 잘하는 누군가에게 일방적으로 의존하면서 최종 답을 받아내는 기술을 의미하는 것은 더더욱 아니다. 가르쳐주는 사람에게 적절히 의존하면서 스스로 발전해가는 모습을 보여주어 상대가 보람과 자부심을 느낄 수 있도록 하는 소통 감각이 필요하다.

언어에는 문법과 논리적 규칙이 있다. 한편 우연한 규칙이나 예외들로 가득 차 있고 예측하기 힘든 방향으로 진화해간다. 이는 언어의 규칙을 자연스럽게 습득하고는 있으나 그 규칙의 보존에는 무심한 사용자들이 집합적으로 언어를 사용하기 때문이다. 프로그래밍 언어도 논리적 문법에 기반해 설계되나, 그를 이용해 코딩하는 사람들이 증가하고 코딩 내용이 공유될수록 최초 설계자는 예상하지 못한 방향으로 진화되어 간다. 전문 프로그래머가 되기 위해서가 아니라 소통 언어로서 코딩을 익히려 한다면, 꽉 짜여진 설계에 근거한 체계를 익히고 정답을 찾아간다기보다는 역동적인 사용자로서 언어의 진화에 동참한다는 자세로 코딩을 익히는 것이 효과적일 것이다.